

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI  
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**Sistem de management al centrului de transfer tehnologic iTransfer**

propusă de

***Stoian Ioan-Cătălin***

**Sesiunea: Iulie, 2019**

**Coordonator științific**  
**Conferențiar Doctor Lenuța Alboaic**

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**  
**FACULTATEA DE INFORMATICĂ**

**Sistem de management al centrului de transfer tehnologic**  
**iTransfer**

***Stoian Ioan-Cătălin***

**Sesiunea:** Iulie, 2019

**Coordonator științific**  
**Conferențiar Doctor Lenuța Alboaie**

**Avizat,  
Îndrumător Lucrare**

Titlul, Numele și prenumele

Data \_\_\_\_\_

Semnătura \_\_\_\_\_

**DECLARAȚIE**  
**privind originalitatea conținutului lucrării de licență**

Subsemnatul Stoian Ioan-Cătălin

domiciliul în Jud. Galați, Loc. Galați, Str. Lozoveni Nr. 13, născut la data de 23.02.1996 , identificat prin CNP 1960223170061, absolvent al Universității „Alexandru Ioan Cuza” din Iași, Facultatea de Informatică specializarea Informatică, promoția 2018,

declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul:

**Sistem de management al centrului de transfer tehnologic iTransfer**

elaborată sub îndrumarea Conferențiar Doctor Lenuța Alboaie, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență/disertație să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data azi, .....

Semnătură student .....

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „***Sistem de management al centrului de transfer tehnologic iTransfer***”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

---

Absolvent *Stoian Ioan-Cătălin*

(semnătura în original)

## Cuprins

I.	Introducere .....	5
	I. 1 Motivație .....	5
	I. 2 Utilitate .....	6
II.	Arhitectura sistemului .....	8
	II. 1 Interfața vizuală .....	8
	II. 2 Aplicația server .....	12
	II. 3 Schema bazei de date .....	15
	II. 4 Securitate .....	17
	II. 5 Integrare cu 3rd-parts (GMail, GCalendar) .....	18
III.	Cazuri de utilizare .....	20
	III. 1 Utilizatori normali .....	20
	III. 2 Utilizatori administrativi .....	27
	III. 3 Alte cazuri de utilizare .....	35
IV.	Concluzii .....	37
	IV. 1 Rezumat .....	37
	IV. 2 Direcții viitoare .....	37
V.	Bibliografie .....	38
VI.	Anexa 1. Tehnologii utilizate .....	39
VII.	Anexa 2. Securitate, Autentificare si QR code .....	41
VIII.	Anexa 3. Integrare cu Google API .....	42

## **I. INTRODUCERE**

Aplicația **Sistem de management al centrului de transfer tehnologic iTransfer** denumită în continuare **iTm** este un software de management a spațiilor de lucru în echipă (co-working spaces), conceput astfel încât să se muleze pe cazurile de utilizare și nevoile centrului de transfer tehnologic iTransfer.

Software-ul de management este cel care are rolul de a raționaliza și automatiza procesele de management pentru a diminua complexitatea proiectelor și sarcinilor mari, precum și pentru a încuraja sau facilita cooperarea în echipă, colaborarea și raportarea corespunzătoare a proiectelor.

### **I.1 Motivație**

Centrul de transfer tehnologic iTransfer este un program destinat atât studenților ce vor să pornească un start-up, cât și profesorilor sau specialiștilor IT&C din domeniu ce vor să devină independenți dezvoltându-și propriile idei inovatoare. Acesta își propune să îi ajute pe aceștia din urmă, oferindu-le atât un spațiu de lucru complet echipat, cât și acces la diferite evenimente și conferințe din domeniul IT&C.

Unui astfel de program, îi este necesar un software care să implementeze și să ofere centrului o posibilitate ușoară de a funcționa, fără prea multe procese birocratice. În secolul XXI tendința este din ce în ce mai mare să se renunțe la birocrație și să fie înlocuită cu programe inteligente care pot genera automat toate documentele cerute de legea în vigoare. Tot mai multe instituții din întreaga lume apelează la astfel de software-uri pentru că nu doar face munca angajaților mai ușoară, dar s-a observat că o instituție / program ce beneficiază de un software inteligent ce se ocupă cu administrarea proceselor interne și externe, automat crește și satisfacția clienților sau a persoanelor ce interacționează cu el.

Entitățile care folosesc software-uri pentru desfășurarea activităților beneficiază nu doar de simplificarea tuturor proceselor cât și de beneficii care nu ar fi posibile în lipsa unui astfel de "tool", precum notificările automate, un istoric generat automat sau posibilitatea de a căuta informațiile dorite mult mai ușor.

Într-un secol în care toată lumea are un telefon mobil cu acces la internet, ori o casuță poștală digitală (email), ori un cont pe o rețea de socializare, unde totul este digitalizat iar oamenii știu să se descurce de la cele mai fragede vârste cu aceste tehnologii, se impune automat și adoptarea de software-uri în cadrul instituțiilor, persoanelor juridice sau

programelor de orice natura. Astfel oamenii se pot concentra doar pe ceea ce este cu adevărat important, pe ceea ce necesită creativitatea și înțelepciunea ființei umane, lăsând roboților părțile repetitive, lucrurile ce nu necesită rațiune. O astfel de abordare cu siguranță poate crește productivitatea creând în jurul nostru un mediu mult mai plăcut de dezvoltare.

## **I.2 Utilitate**

Aplicația **iTm** facilitează procesele din interiorul centrului de transfer tehnologic iTransfer, punând la dispoziție un software gata să administreze fiecare resursă, acțiune sau membru înregistrat și să ofere o interfață ușor de utilizat atât pentru membrii administrativi cat și clienților centrului.

De această aplicație vor beneficia următoarele categorii:

- Personalul administrativ al centrului
- Studenții
- Profesorii
- Specialiști IT&C din domeniu

Chiar dacă **iTm** a fost conceput special pentru o anumită entitate, generalitatea cu care a fost proiectat permite folosirea lui pentru orice spațiu de lucru din categoria ”co-working”.

Pentru ca aplicația să fie ușor de utilizat, canalul principal de comunicare cu utilizatorul, interfața, a fost proiectată astfel încât să respecte regulile descrise de Abhishek Kumar in articolul [UI Design Guide 2018 - Rules of User Interface Design](#).

Software-ul dorește să încorporeze toate procesele necesare, iar acestea fiind următoarele:

- Centralizarea și administrarea participanților. Acest lucru se face prin reprezentarea fiecărui membru / echipe printr-un cont pe aplicație, unde sunt reținute toate informațiile necesare precum date personale, contract în forma scanată, email, număr telefon, spațiul de lucru asignat, chitanțele scanate de la plăților precedente.
- Facilitarea efectuării plăților prin intermediul ordinului de plată bancar sau plata cu cardul efectuată online.
- Planificarea evenimentelor și a conferințelor de către personalul administrativ dar și administrarea participanților din rândul membrilor **iTm**.

- Distribuirea membrilor în spațiile de lucru oferite sub forma unei hărți interactive ce ilustrează exact cum sunt așezate acestea. Utilizatorii pot selecta spațiul pe care îl doresc vizualizând în timp real care sunt cele disponibile, care sunt cele ocupate sau rezervate și unde sunt poziționate.
- Logarea intrărilor în spațiile de lucru a membrilor, memorand data și ora la care sa întâmplat. Accesul în clădire se face pe baza unui QR Code ce trebuie scanat la intrare.
- Notificarea prin email a persoanelor responsabile. Câteva dintre acestea sunt: notificarea secretarilor atunci când exista un nou client, notificarea contabililor când se efectuează o noua plată (în cazul plății prin ordin bancar sau cu cardul), notificarea clientului că plata a fost înregistrată (în cazul plății direct la sediul iTransfer), notificarea clientului când o nouă factură a fost generată de către sistem.
- Generarea de rapoarte în scopul contabilității pe care pot fi aplicate filtre pentru a obține exact rapoartele dorite, mai apoi existând posibilitatea de a fi descărcate în format excel.
- Posibilitatea de a planifica ședințe în spațiile special amenajate pentru acest lucru. Acestea ședințe sunt sincronizate cu google calendar, astfel ele o sa apară în calendarul utilizatorilor.

În comparație cu metodele clasice de administrare, folosirea unei aplicații de acest tip câștiga cu brio, oferind o multitudine de beneficii tuturor celor care iau parte. Birocrația devine aproape inexistentă, multe dintre procese devin automatizate, mult mai eficiente și practice atât pentru personalul administrativ cât și pentru membrii centrului iTransfer.



## II. ARHITECTURA SISTEMULUI

În acest capitol voi vorbi despre cum este structurată și gândita arhitectura întregii aplicații, de la cele mai înalte nivele până la snapshot-uri de cod.

Putem afirma că aplicația este alcătuită din două componente independente dar care colaborează pentru a forma produsul final, acestea fiind aplicația server și interfața **vizuală**. Datorită faptului că cele două componente au scopuri diferite, acest lucru impune o arhitectură diferită pentru fiecare împarte. Arhitectura generală a aplicației arată astfel:

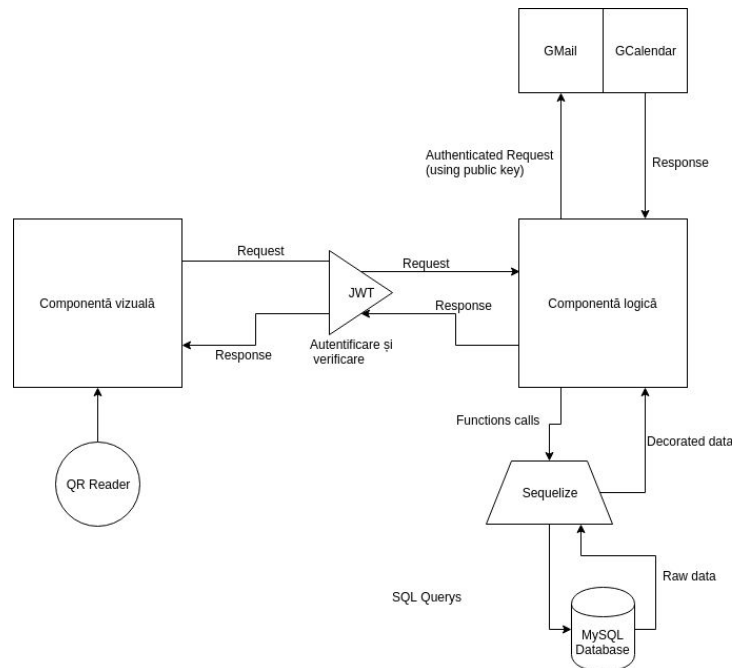


Figura 1 - Arhitectura generală

### II.1 Interfața vizuală

Interfața vizuală este un MVC (model-view-controller) ce integrează redux pattern (model) dezvoltat de Facebook pentru aplicațiile de frontend de acest tip. De fapt acest pattern este în esență un superset peste **State pattern** care este un software design pattern comportamental (behavior). Acesta se folosește de 3 resurse principale: actions, reducers și store, cu scopul de a manipula starea aplicației.

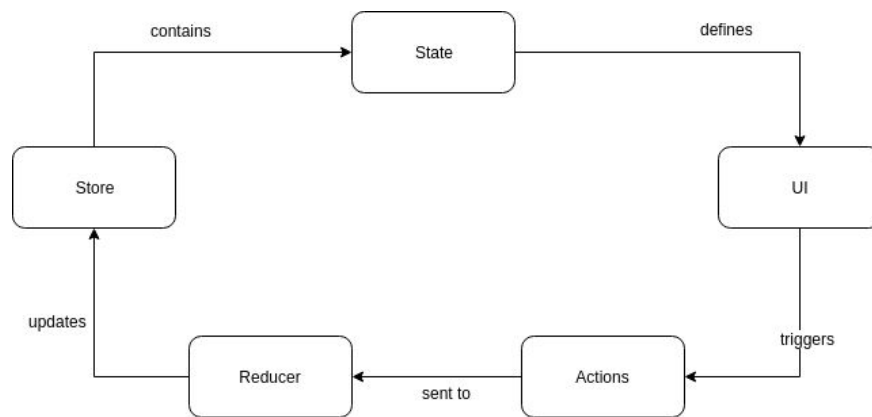


Figura 2 - Redux pattern

Aplicația are nevoie de o stare inițială și definită în prealabil care să explice ce forma vor avea datele. Această stare este reprezentată sub formă de JSON. Această este modificată pe parcursul utilizării aplicației de către anumiți declanșatori activați de către utilizator. Acești declanșatori generează ”acțiuni”. Prin intermediul acțiunilor ”reducer”-ii știu cum să altereze starea aplicației pentru a obține rezultatul dorit. Mai multe detalii despre aceste tehnologii se pot regăsi în Anexa 1.

```

21 export function userReducer(state = initialState, action) {
22   let newState;
23   switch (action.type) {
24     case LOGIN_USER:
25       newState = { ...state, ...action.payload, authenticated: true };
26       sessionStorage.setItem('user', JSON.stringify(newState));
27       if (newState.token) {
28         axios.defaults.headers.common['Authorization'] = `Bearer ${newState.token}`;
29       }
30       return newState;
31     case LOGOUT_USER:
32       sessionStorage.removeItem('user');
33       return {
34         full_name: '',
35         role: '',
36         email: '',
37         password: '',
38         token: '',
39         authenticated: false,
40         members: [],
41         plan: null,
42       };
43     case SET_MEMBERS:
44       newState = { ...state, members: action.payload };
45       sessionStorage.setItem('user', JSON.stringify(newState));
46       return newState;
47     default:
48       return state;
49   }
50 }

```

Figura 3 - Exemplu de reducer implementat

Odată ce starea aplicației a fost modificată, ea este trimisă în ”store” de unde fiecare element de UI extrag părțile de interes specifice pentru a-și adapta proprietățile.

Componenta are structurate elementele de UI în mod piramidal, astfel încât se regăsesc: elemente atomice precum tag-uri native de html (div, span, p) cât și tag-uri personalizate (DatePicker, TableCell, Table, IconButton), componente&layout-uri (reprezentate de o colecție de elemente atomice) și pagini ce sunt alcătuite din aceste componente.

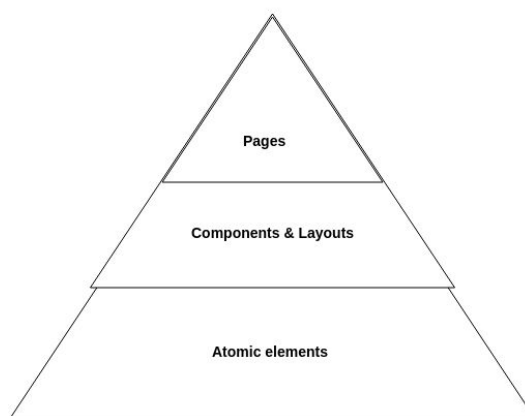


Figura 4 - Piramida structurii elementelor din interfața vizuală

**Navigarea** în interiorul aplicației vizuale este asigurată de API-ul ”history” implementat în fiecare browser modern. Acesta este reprezentat de o stivă unde adresele sunt adăugate sau scoase în funcție de acțiunile celui ce folosește aplicația. Acest API este controlat de un element atomic numit ”Route”, element care decide dacă afișează sau nu paginile în funcție de locația unde se află utilizatorul. De fapt, utilizatorul niciodată nu schimbă locația efectiv, ea este mereu aceeași, însă locațiile din interiorul stivei și paginile sunt modificate în mod dinamic, încărcându-se doar anumite părți ale aplicației.

Deoarece aplicația este una restricționată de o autentificare prealabilă, nu toate componentele sunt accesibile, unele dintre ele necesitând această stare de autentificare. Pentru a rezolva această problemă am despărțit elementul Route în două subcategorii: **PublicRoute** și **PrivateRoute**. Cele două funcționează în oglindă, adică dacă starea aplicației este autentificată atunci PrivateRoute va permite accesul locațiilor din cadrul ei iar PublicRoute nu. Altfel, se va întâmpla exact invers.

```

5 class PrivateRoute extends React.Component {
6   render() {
7     if (this.props.authenticated) {
8       return <Route {...this.props} />;
9     } else {
10      return <Redirect
11        to={{
12          pathname: "/login",
13          state: { from: this.props.location }
14        }}
15      />
16    }
17  }
18 }

```

```

5 class PublicRoute extends React.Component {
6   render() {
7     if (!this.props.authenticated) {
8       return <Route {...this.props} />;
9     } else {
10      return <Redirect
11        to={{
12          pathname: "/",
13          state: { from: this.props.location }
14        }}
15      />
16    }
17  }
18 }
19 }

```

Figura 5 - Codul celor două componente atomice PrivateRoute și PublicRoute

Însă nivelul vizual nu înseamnă numai logică de interacțiune cu utilizatorul. O bună parte din ea este reprezentată și de partea de design, unde regăsim elemente precum: layout, elemente și cromatică, flow-uri intuitive.

Layout-ul este modul cum este divizată aplicația pentru a deveni cât mai intuitivă și mai plăcută ochiului uman. Indiferent de cum este împărțit, există patru categorii care trebuie să se regasească, și anume: antete (headers), meniuri, conținut și subsoluri (footers). Aplicația dezvoltată de mine le conține pe toate cele 4. Pentru a ajunge la forma finală, prima dată au fost proiectate (desenate) într-un program specializat pe acest lucru, cum ar fi Photoshop. Odată ce layout-ul a fost gata, a trebuit să desenez forma elementelor, modul cum acestea vor arăta și detaliile lor, precum culoarea de background, border-ul, margins&padding, etc. În acest pas a trebuit să urmăresc un color pattern prestabilit de mine. Acesta este alcătuit din culori precum albastru, culoare principală, roz culoare secundară și gri, culoare de ambient. Când am avut tot acest proces gata, și a trebuit să transpun tot desenul în cod, tot atunci am proiectat și flow-urile aplicației. Acest lucru înseamnă că am avut în vedere ca animațiile să nu distragă utilizatorul de la sarcinile principale, elementele să ofere un confort maxim în a trece de la un proces la altul, iar cea mai grea parte, din punctul meu de vedere, a fost să proiectez întreaga aplicație conform principiului 3 clicks rule. Această regulă spune că utilizatorul trebuie să fie capabil să ajungă oriunde din maxim 3 click-uri.

Stilul de design pe care l-am ales este "material design" iar pentru a nu reinventa roata și a mă putea concentra pe flow-urile aplicației, am ales să folosesc [material-ui](#). Această bibliotecă îți permite să folosești elemente atomice precum butoane, modale, tabele deja scrise în stilul material. Eu a trebuit doar să adaptez aceste elemente la design-ul aplicației mele pentru a se potrivi.

## II.2 Aplicația server

Arhitectura acestei componente este proiectată ca un RESTfull API, care pune la dispoziția aplicației client 4 resurse: Users, Payments, Events, Logs, Plans. Acestor resurse le sunt atribuite câte un cuvânt cheie sugestiv pentru a fi unic identificate. De exemplu pentru resursa "Event" cuvântul cheie este „/events”. Pe acest punct de acces, se aplică verbele REST pentru a determina ce acțiune se dorește să fie executată.

Resursa "User" reprezintă entitățile pe baza cărora se face autentificarea. Prin intermediul acestei resurse se deosebește ce utilizator folosește aplicația, ce drepturi are în aplicație și ce rute poate accesa. Tot în aceasta resursă se rețin informațiile personale despre utilizator și se construiesc relațiile cu celelalte resurse.

Resursa "Payment" reprezintă facturile și informațiile despre plățile efectuate către centrul de transfer. Se memorează informații precum chitanță, metodă de plată, număr unic de identificare a plății, data plății.

Resursa "Event" reprezintă atât ședințele programate cât și evenimentele organizate de către centrul iTransfer. Aceste evenimente sunt sincronizate cu google calendar.

Resursele "Plans" și "Logs" reprezintă planurile pe care utilizatorii și le pot alege, respectiv log-urile aplicației. Log-urile memorează lucruri precum accesul în spațiile de lucru.

Informația în interiorul acestei componente are un flow bine definit de 3 tipuri de fișiere: Controllers, Commands, Models. Un **controller** este locul unde se definesc căile de acces către acțiunii. Fiecare acțiune nu trebuie să aibe decât câteva linii de cod, fiind relativ mici. În general acestea doar determină status-ul și body-ul cu care să răspundă:

```
44 router.post('/meeting', isAuthenticated, async (ctx, next) => {
45   const meetingBody = ctx.request.body;
46   const eventsCommands = new EventsCommands();
47   const response = await eventsCommands.addMeetings(meetingBody, ctx.state.user);
48   ctx.response.body = response;
49 });
```

Figura 6 - Exemplu de rută dintr-un controller

O comandă (command) este locația unde se află business logicul aplicației. Aici sunt scriși algoritmi ce prelucrează informațiile primite de la client. Aceste comenzi sunt împărțite pe resurse, astfel sa fie cât mai intuitiv atunci când cineva dorește să facă debugging. Comenzile trebuie sa conțină numai logică și prelucrare de informații. Fiecare funcție va returna mereu un răspuns și un status code conform standardelor RESTful API. Acestea din urmă, vor fi returnate de către controllere clienților care au făcut API call-ul.

```

34  async deleteFileAndUncompletePayment(paymentId, userId) {
35      const payment = await Payment.findOne({
36          where: { id: paymentId }
37      });
38      if (payment.userId !== userId) {
39          return {
40              status: 401,
41          }
42      }
43      fs.unlink(payment.recipeUrl, () => {
44          console.log('File deleted...');
45      });
46      payment.payedDate = null;
47      payment.status = 'unpaid';
48      payment.recipeUrl = null;
49      await payment.save();
50      return {
51          status: 200,
52      }
53  }

```

Figura 7 - Exemplu de funcție comandă

În exemplul de mai sus, avem o funcție din class comandă-ul "PaymentCommand". Scopul funcției este să invalideze (să șteargă) o plată care a fost marcată anterior ca și achitată. Această funcție primește ca și parametrii id-ul unic intern al plății și id-ul unic al utilizatorului care dorește să efectueze această acțiune. Prima dată se caută în baza de date dacă plata chiar există. Dacă răspunsul este afirmativ atunci se verifică user-ul care dorește să efectueze această acțiune chiar are acces să o facă. În cazul unui răspuns negativ, funcția va returna status-ul 401, specific răspunsurilor pentru cererile neautorizate. În cazul în care valoarea variabilei *userId* este egal cu id-ul utilizatorului care deține plata, atunci plata va fi invalidată astfel: Chitanța va fi ștersă din sistem, Data la care s-a făcut plata va deveni null iar status-ul va fi setat cu valoarea "unpaid".

**Modele** sunt fișiere ce conțin o singură clasă, fiecare dintre acestea reprezentând o tabelă a bazei de date.

```

1  const Sequelize = require('sequelize');
2
3  const Meeting = (sequelize, type) => {
4    return sequelize.define('meeting', {
5      id: {
6        type: Sequelize.INTEGER,
7        primaryKey: true,
8        autoIncrement: true
9      },
10     name: {
11       type: Sequelize.STRING,
12       allowNull: false
13     },
14     startDate: {
15       type: Sequelize.BIGINT,
16     },
17     endDate: {
18       type: Sequelize.BIGINT,
19     },
20     attenders: {
21       type: Sequelize.TEXT,
22     },
23     gcalendar_meeting_id: {
24       type: Sequelize.STRING,
25     }
26   })
27 };
28
29 module.exports = Meeting;

```

Figura 8 - Exemplu de model a tabelii Meeting

Auxiliar acestei structuri, se adaugă alte 2 tipuri, middlewares și jobs. Middleware-urile sunt ”porțile” prin care trece request-ul înainte să ajungă la controllere. Aceste porți au diferite roluri, precum verificarea stării de autentificare sau verificarea dacă există suficiente drepturi.

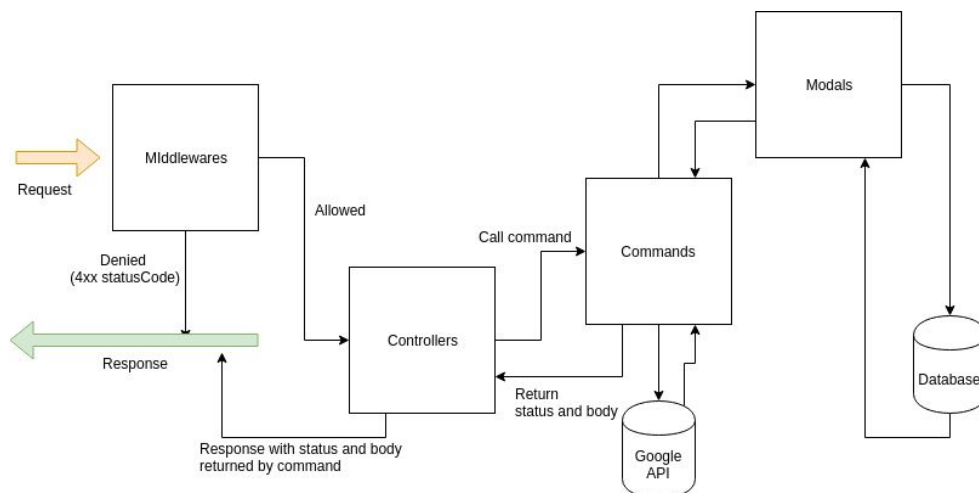


Figure 8 - Parcursul unui request către aplicația server

Pentru a face posibilă afișarea în timp real a utilizatorilor care accesează spațiile de lucru utilizând QR code-ul unic atribuit fiecărui utilizator, s-a folosit protocolul Websocket pentru comunicarea între doua sau mai multe browsere deschise. Această comunicare se petrece pe aplicația vizuală, însă un rol foarte important este jucat de aplicația logică, ce

intermediază acest proces. Mai multe detalii despre tehnologiile folosite pot fi regăsite în Anexa 1.

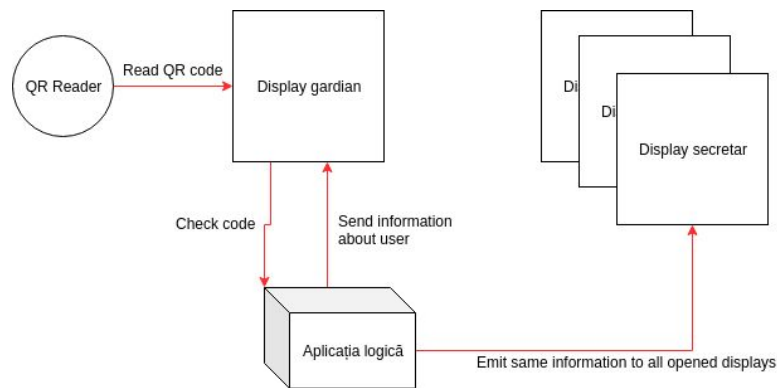


Figura 10 - Folosirea websocket-urilor în aplicație

## II. 3 Schema bazei de date

Baza de date este memoria persistentă a aplicației, ea reprezentând totodată și starea actuală a aplicației. Pentru acest proiect am folosit o bază de date rațională, datorită faptului că între informațiile memorate exista relații naturale, este nevoie de o reprezentare bidimensională și de implementarea anumitor constrângeri.

Pentru automatizarea query-urilor efectuate către baza de date am folosit un ORM (Object-relational mapping) numit Sequelize. Acesta este unul specific limbajului NodeJS.

Un ORM este o tehnică în limbajele de programare pentru a ilustra bazele de date sub forma obiectelor din programarea orientată pe obiect, clase. Fiecare clasă reprezintă o tabela din baza de date, iar fiecare instanțiere a clasei respective reprezintă un rând din tabela respectivă. Mai multe detalii pot fi regăsite în Anexa 1.

Baza de date este alcătuită din 11 tabele după cum urmează:

- **Users:** În această tabelă se află atât utilizatorii normali cât și cei administrativi. Diferența dintre cele două tipuri o face coloana "role". Această coloană poate avea valori de la 0 la 3, unde 0 înseamnă utilizator simplu, iar 1,2,3 sunt roluri de administrare. (e.g. Contabil, Secretar, Administrator).
- **Offices:** Este o tabelă în care se află spațiile de lucru care pot fi închiriate de utilizatori.
- **Members:** Conține membrii adiționali ai conturilor normale.
- **Plans:** Este o tabelă statică, adică ea nu își schimbă conținutul foarte des deoarece conține planurile de beneficii puse la dispoziție de centrul iTransfer.



- Facilities: Conține beneficiile asociate fiecărui plan.
- Payments: În această tabelă se stochează toate plățile a tuturor clienților. Aceasta are date importante și sensitive despre plățile efectuate precum locația pe disk a chitanțelor, data când a fost achitată plata sau metoda de plată cu id-ul tranzacției în cazul plății cu cardul.
- Meetings: Aici se rețin date despre meeting-urile programate de utilizatori. Câmpul `gcalendar_meeting_id` reține id-ul evenimentului adăugat în google calendar cu scopul de a ști ce eveniment trebuie șters în cazul anulării meeting-ului.
- Rooms: Această tabelă este la fel una statică deoarece aici se rețin informații despre sălile unde utilizatorii își pot programa ședințele.
- Events: Această tabelă reține informații despre evenimentele create în cadrul iTransfer.
- Attenders: Este strict legată de tabela anterioară, ea memorând utilizatorii care s-au înscris la evenimente.
- OfficeAccesses: În această tabelă se rețin log-urile pentru intrările în spațiile de lucru a utilizatorilor.

Schema bazei de date ce reprezintă relațiile dintre tabele și toate atributele lor este următoarea:

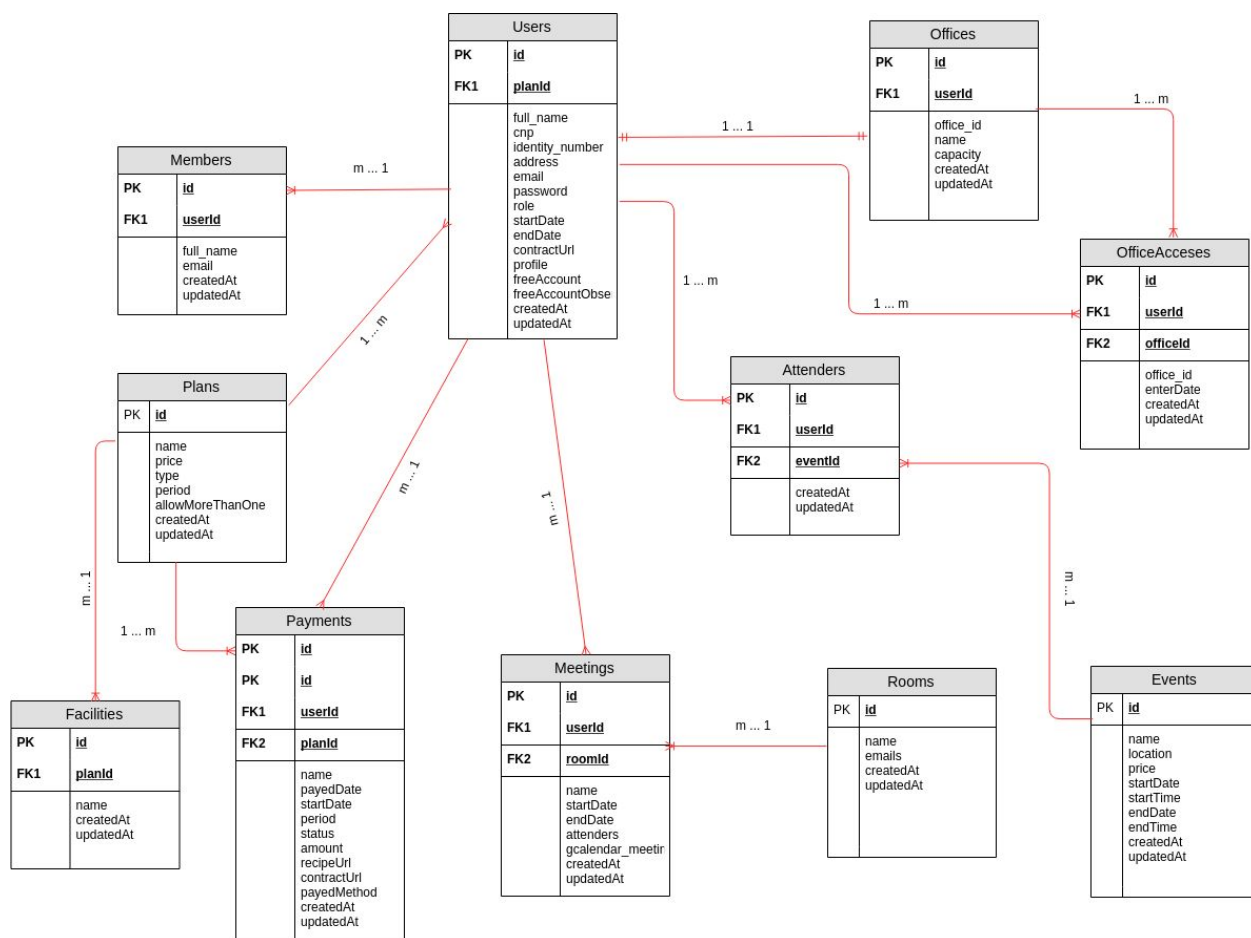


Figura 11 - Schema bazei de date

Această bază de date este una normalizată deoarece respectă regulile de baza a normalizării și anume: fiecare coloană dintr-un tabel conține o singură valoare de un singur tip, fiecare tabel are un identificator unic, nu sunt grupuri de date repetitive și nu există posibilitatea de a avea două linii identice în tabel.

## II. 4 Securitate

Securitatea comunicării dintre componentele aplicației este asigurată de un protocol numit **JsonWebToken** (JWT). Spre deosebire de modul clasic de a identifica un utilizator prin sesiuni web, acest nou sistem permite ca componentele aplicației să fie complet independente între ele, astfel încât să poată exista una fără alta.

Metoda JWT se folosește de token-uri ce sunt atașate în header-ul fiecărui request, astfel încât aplicația care procesează request-ul să identifice autorul. Aceste token-uri sunt de încredere deoarece ele sunt semnate digital. Asta înseamnă că peste un obiect JSON ce

conține datele unui utilizator, se aplică o semnătură digitală folosind algoritmi precum **HMAC** (folosind un singur secret) ori **RSA** sau **ECDSA** (folosind o cheie publică și una privată). Întreaga metodă este descrisă de standardul [RFC 7519](#).

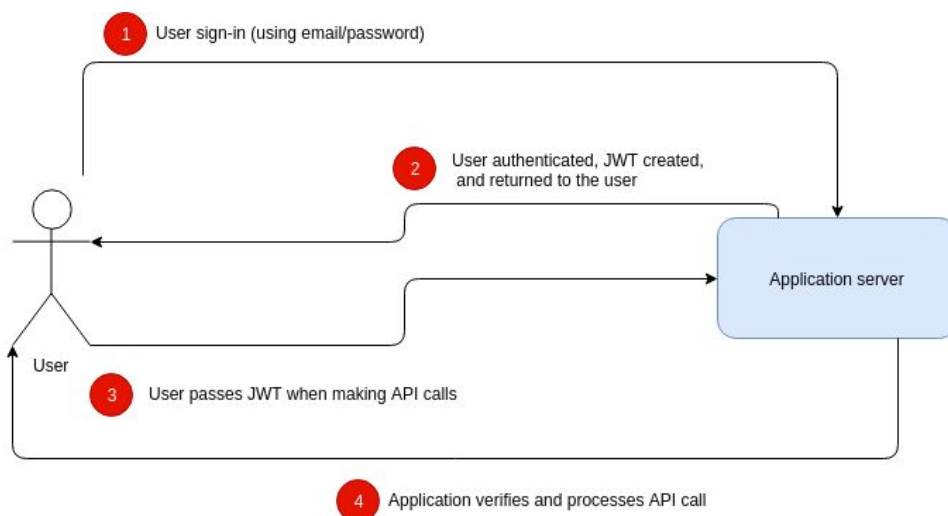


Figure 12 - JWT flow

Deoarece sistemul impune ca unii utilizatori să poată executa anumite chestii iar alții nu, am implementat un sistem de roluri, astfel să reușesc să stabilesc ce permisiuni are fiecare cont. Sistemul de roluri pune la dispoziție 4 tipuri, acestea fiind: Normal, Contabil, Secretar, Admin. Restricțiile de acces pe API este implementată de 3 middleware-uri ce verifică dacă autorul are sau nu acces.

Pentru a asigura integritatea sistemului de loguri pentru intrările în sălile de lucru, am dezvoltat un sistem de securitate bazat pe parola și cheie publică. Acesta funcționează astfel: Parola este tipărită pe o legitimație sub forma unui cod QR. Pe pagina destinată scannerului, acesta nu v-a funcționa fără să se scaneze prima dată parola. Atât timp cât parola nu a fost scanată, nici o altă scanare nu v-a funcționa. Odată ce parola a fost scanată, aplicația server întoarce către client o cheie publică pe care acesta o va folosi la fiecare scanare pentru a fi recunoscut de către server ca fiind de încredere.

Mai multe detalii despre JWT se pot regăsi în Anexa 2.

## II. 5 Integrare cu 3rd-parts (GMail, GCalendar)

Integrarea cu 3rd-parts precum GMail și GCalendar a fost necesară datorită nevoii de a putea sincroniza activitățile din interiorul centrului de transfer iTransfer cu email-urile personale ale utilizatorilor. Această integrare se folosește de API-ul expus de Google și sunt necesari o anumită serie de pași pentru a stabili un canal de comunicare sigur între **iTm**.

Pentru a ne asigura că toate request-urile sunt făcute chiar de aplicația **iTm**, aceasta trebuie să efectueze o serie de pași în scopul autentificării. Se folosește un cont de google la care se adaugă un cont de sistem. Acest tip de cont poate să acceseze conturile altor utilizatori de la care primește aceste drepturi. Comunicarea se face folosind JWT cu RSA, motiv pentru care avem nevoie de o cheie privată și una publică.

```
65     let jwtClient = new google.auth.JWT(  
66         privatekey.client_email,  
67         null,  
68         privatekey.private_key,  
69         [  
70             'https://www.googleapis.com/auth/calendar',  
71         ]);  
72     jwtClient.authorize(function (err, tokens) {
```

Figura 13 - Exemplu cod pentru autentificare către Google

Odată cu inițializarea, trebuie să se ofere și scope-uri, niște string-uri care specifică către google ce dorim să facem în continuare cu API-ul lor. Acesta ne va oferi acces doar la acele părți de API ce au fost precizate în scope.

Odată ce autentificarea este gata, aplicația folosește biblioteca *googleapis* pentru a face api call-urile necesare.

### III. CAZURI DE UTILIZARE

În acest capitol voi descrie modul cum utilizatorii pot interacționa cu aplicația. Pentru că aplicația poate fi folosită de două categorii de utilizatori, voi împărți de asemenea acest capitol în două sub-capitole: Primul destinat utilizatorilor normali, iar cel de-al doilea destinat utilizatorilor administrativi. Pentru a explica cât mai bine mă voi folosi atât de screenshot-uri ale aplicației cât și de diagrame de tipul "use-case".

Aplicația este structurată astfel încât prima dată ți se afișează o pagină de autentificare. Pentru a părăsi această pagină ești obligat să te autentifici cu contul de utilizator, ori să mergi pe pagina de înregistrare unde îți poți crea un cont nou.

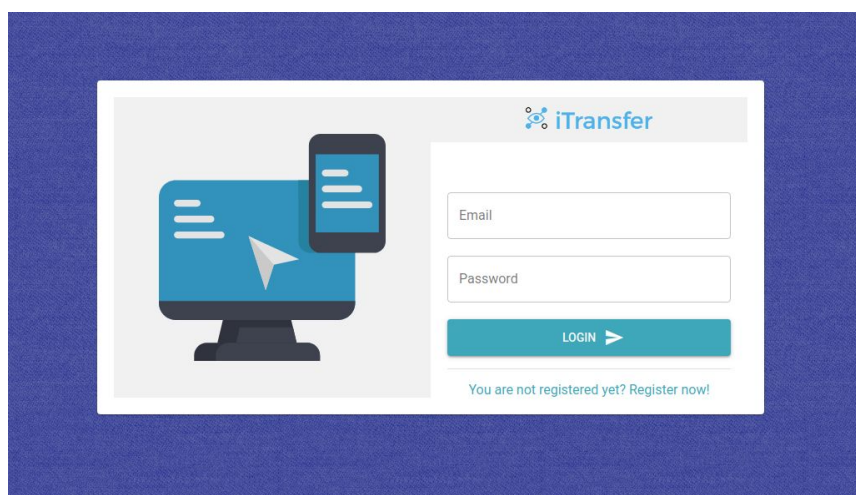


Figura 14 - Pagina destinată autentificării

În această pagină se află două input-uri, unul pentru email și altul pentru parolă. Pentru autentificare trebuie apăsat butonul "Login" sau tasta "Enter". În cazul în care utilizatorul introduce un email sau o parolă greșită, acesta va fi anunțat de un snackbar ce va conține descrierea erorii.

#### III. 1 Utilizatori normali

Pentru utilizatorii normali, care încă nu au un cont, își pot crea unul apăsând text-ul *"You are not registered yet? Register now!"*. Aceștia vor fi redirecționați către pagina de înregistrare:

Figura 15 - Pagina pentru înregistrare și pașii acestui proces

Așa cum se observă în poza de mai sus, utilizatorii au de completat patru pași simpli în care să ofere date despre: subscription-ul plan-ul pe care îl doresc, date personale și spațiul de lucru pentru care optează. În funcție de tipul plan-ului ales aceștia vor trebui să ofere date despre o singură persoană (pentru plan-urile de tipul single) sau pentru întreaga echipă (pentru plan-urile de tip team access). Date precum CNP, Serie, Adresa se cere doar pentru liderul echipei. În vederea alegerii spațiului de lucru, aceștia vor vizualiza o hartă a clădirii, împărțită în spațiile de lucru și reflectând așezarea reală în spațiu a acestora. Acest proces se finalizează prin apăsarea butonului "Finish". Dacă toate datele sunt validate, aceștia vor primi un mesaj în care sunt notificați că trebuie să aștepte confirmarea de la personalul administrativ. Aceștia nu vor fi capabili să acceseze contul până acesta nu va fi validat. Odată cu crearea contului, personalul administrativ va fi notificat printr-un email de acest lucru.

### Secțiunea Home

Odată ce utilizatorul se autentifică în contul sau, acesta va vedea următorul panou de administrare a contului:

Subscription plan	Status	Date	Bill	Actions
Team Access for students (all team members must be students (400 € / month))	Unpaid	June 10, 2019	-	PAY

Figura 16 - Panoul principal al utilizatorilor normali

De aici utilizatorul poate să rămână pe pagina curentă (vezi figura 16) numită în aplicație și *Home*. Din acest screen utilizatorul poate să își descarce contract-ul în format digital sau să aibe acces la codul QR necesar la intrarea în spațiile de lucru, să administreze membrii echipei și să vizualizeze / efectueze plăți.

Pentru descărcarea contractului, tot ce trebuie să facă este să apese butonul „Download contract” (indicator 1). Ca și mențiune, contul nu devine activ până când contractul nu este urcat în contul utilizatorului.

La indicatorul cu numărul 2 regăsim input-uri pentru editarea email-ului membrilor echipei. Acesta poate fii schimbat doar de liderul echipei.

Plățile aferente contului se regăsesc tot în acest panou, ele fiind indicate de indicatorul cu numărul 3. Toate plățile vor fi afișate în ordine descendentă, afișând numele plan-ului corespunzătoarei lunii pentru care a fost generată plata, status-ul plății (acesta poate fii ”Unpaid”, ”Processing” și ”Paid”), data când s-a generat plata și chitanța / factura (în cazul în care s-a făcut plata).

Pentru a efectua o plată utilizatorul are la dispoziție 3 metode de plată: Plata la sediul iTransfer, plată prin ordin bancar și plata cu cardul. Pentru confirmarea plății la sediul iTransfer se va ocupa personalul administrativ, însă pentru plata prin ordin bancar și plata cu cardul este necesar ca utilizatorul să urmeze câțiva pași simpli:

Pentru **plata prin ordin bancar** utilizatorul trebuie să meargă fizic la bancă, să achite suma de bani necesară, iar apoi să intre pe aplicație iar în panoul de mai sus (figura 16) să apese butonul ”Pay” pentru luna în care dorește să efectueze plata. Imediat după acest pas, i se va afișa următorul modal:

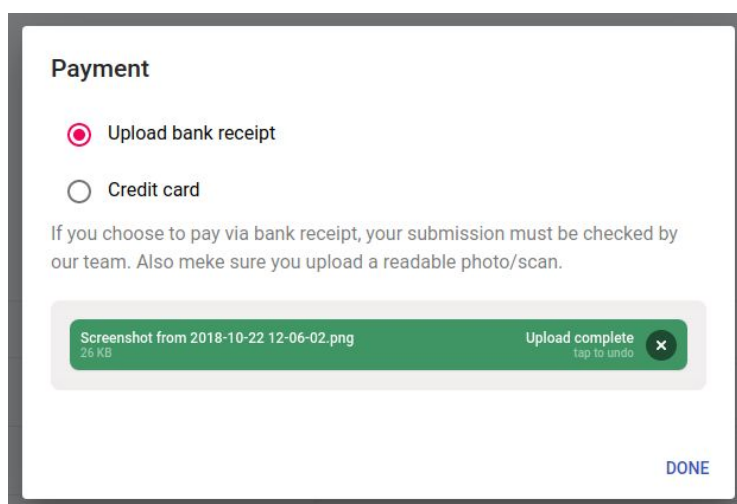
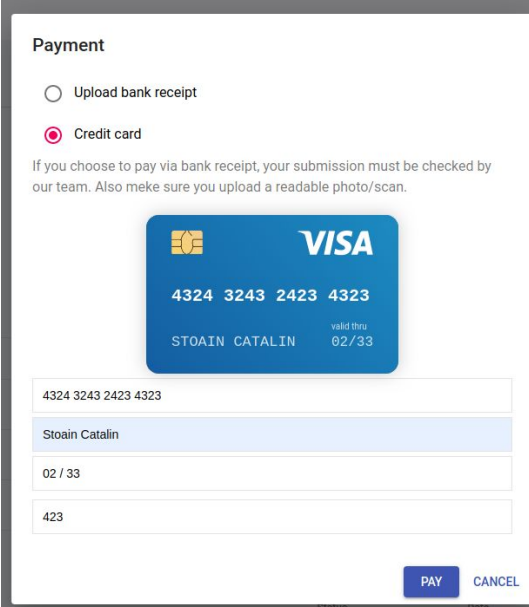


Figura 17 - Modal pentru efectuarea plății prin ordin bancar

Acesta trebuie să urce fișierul făcând click pe aria marcată cu gri și selectând chitanța scanată primită de la bancă, apoi să apese butonul "Done". În caz că a urcat un fișier greșit, poate să revoce acțiunea apăsând butonul "X" din dreptul fișierului. Odată ce a îndeplinit acești pași, plata va trece în status-ul "Processing" până un contabil va confirma plata.

Pentru **plata cu cardul** trebuie urmăriți aceiași pași până la deschiderea modalului. De aici trebuie să selecteze opțiunea "Credit card" și va vedea următorul modal:

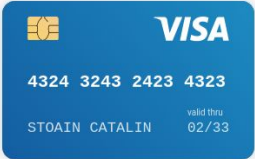


Payment

☐ Upload bank receipt

☒ Credit card

If you choose to pay via bank receipt, your submission must be checked by our team. Also make sure you upload a readable photo/scan.



4324 3243 2423 4323

Stoain Catalin

02 / 33

423

PAY CANCEL

Figura 18 - Modal pentru efectuarea plății cu cardul

În acest modal utilizatorul trebuie să completeze câmpurile aferente cu datele cardului de pe care dorește să efectueze plata, iar pentru a finaliza procesul este necesar apăsarea butonului "Pay". Odată ce plata este procesată status-ul acesteia va fi "Paid" și va primi un id unic de plată ce se va regăsi și în contul bancar.

### Secțiunea Meetings

O altă pagină pe care utilizatorii o pot accesa este aceea unde își pot programa și rezerva întâlnirile în sălile puse la dispoziție de centru.



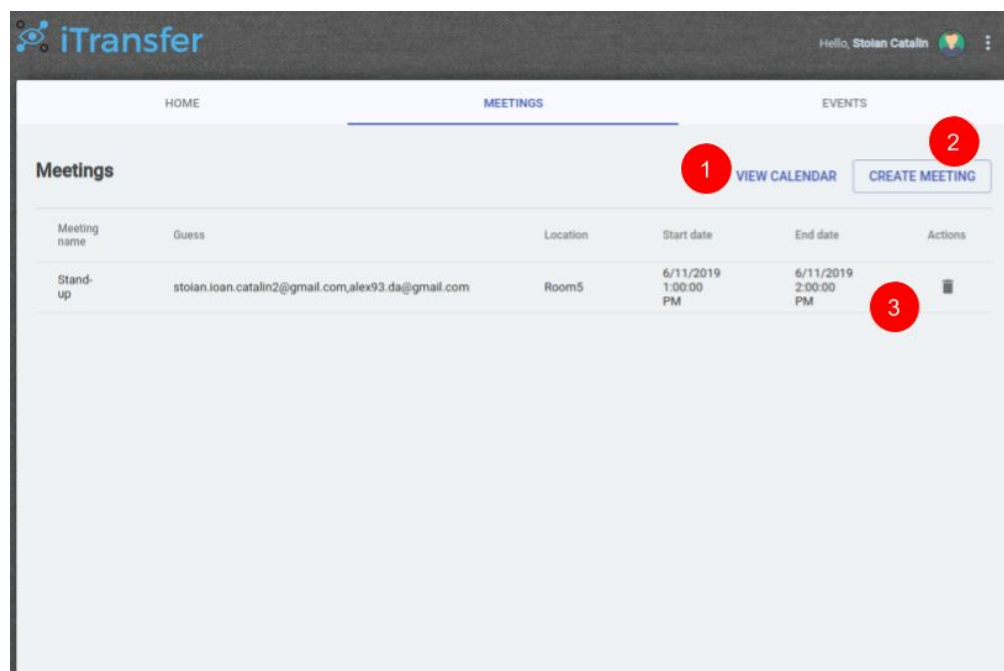


Figura 19 - Pagină pentru programarea meeting-urilor

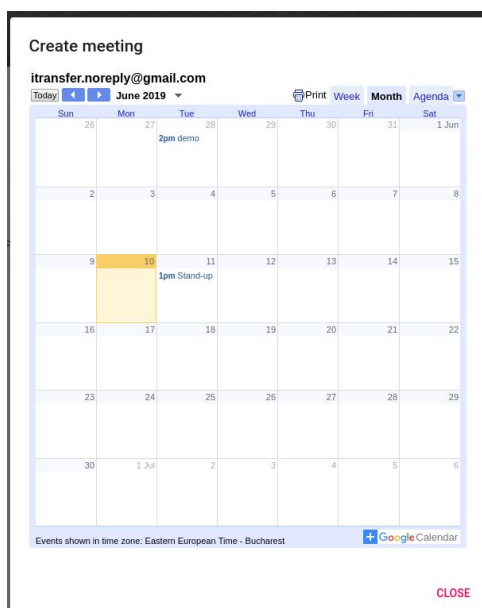


Figura 20 - Calendarul sălilor pentru meeting-uri

În această pagina utilizatorul poate să vadă și să șteargă meeting-urile pe care le are programate. Acesta va vedea informații precum numele meeting-ului, invitații, locația, data când începe și data când se sfârșește. De asemenea când un meeting este creat, acesta va trimite automat invitații către participanți dar și către creator. Meeting-ul se va adăuga automat în google calendar, cu o notificare prestabilită de 10 minute. Pentru a vedea calendarul sălilor utilizatorul trebuie să apese butonul "View calendar" (indicator 1 - figura 19). Acesta va arăta astfel:

Pentru a crea un meeting utilizatorul trebuie să apese butonul "Create meeting" (indicator 2 - figura 19). Odată ce apasă butonul respectiv, următorul buton se va deschide:

The image shows a 'Create meeting' modal window. At the top, it says 'Create meeting' and 'You can invite more people to join this meeting and it will show up in their calendar.' Below this is a text input field for 'Event name' with the value 'Stand Up'. Then, a 'When?' section contains two date-time inputs: 'Start date' (07/03/2019, 02:00 PM) and 'End date' (07/03/2019, 03:00 PM). Below that is a 'Where?' section with a dropdown menu showing 'Room5'. The 'Attendees' section has a list of email addresses: 'stoian.ioan.catalin@gmail.com' and 'catalin.stoian@gmail.com', followed by an empty 'Email' input field. At the bottom right are 'CANCEL' and 'CREATE' buttons.

Figura 21 - Modal pentru crearea meeting-urilor

În acest modal trebuie să completeze formularul cu următoarele date: Numele meeting-ului care va apărea în calendarul tuturor invitaților, data de început și data de sfârșit, locația unde va avea loc (se alege dintr-o listă prestabilită) și invitații, completând cu email-ul acestora, unde vor primi și invitația. Pentru a finaliza procesul utilizatorul trebuie să apese pe butonul "Create".

Pentru a șterge un meeting, trebuie apăsat butonul din dreptul meeting-ului care se dorește să fie șters (indicatorul 3 - figura 19). Un modal de confirmare va apărea pentru a confirma că chiar se dorește ștergerea meeting-ului respectiv.

### Secțiunea Events

Această secțiune este pentru a informa utilizatorii aplicației de viitoarele evenimente în cadrul centrului de transfer informatic, unde se pot înscrie tot de aici. În această secțiune vor apărea evenimentele care urmează să se întâmple.

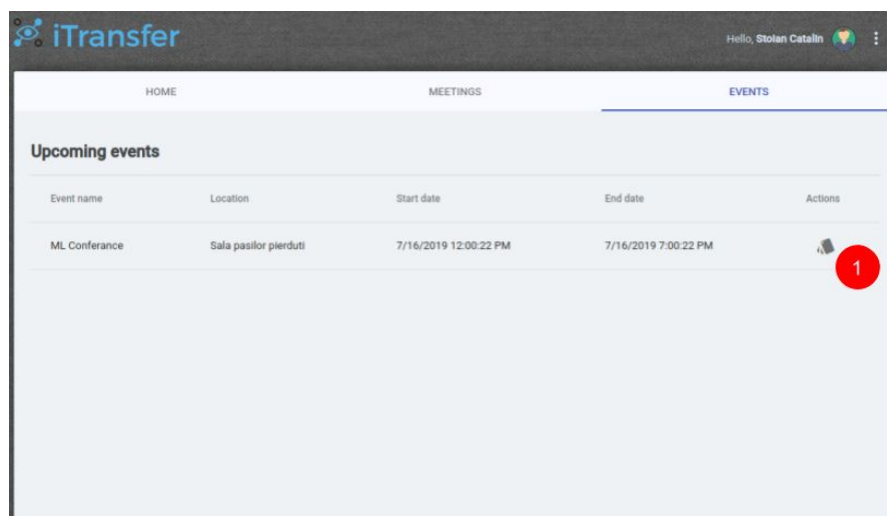


Figura 22 - Secțiunea events

Utilizatorul poate să se înscrie pe el (și restul echipei în cazul în care este abonat la un plan pentru team acces) apăsând butonul din dreapta fiecărui rând al tabelului cu evenimente (indicatorul 1 - figura 22).

Pentru a exemplifica mult mai general toate aceste use-case-uri am creat următoarea diagrama:

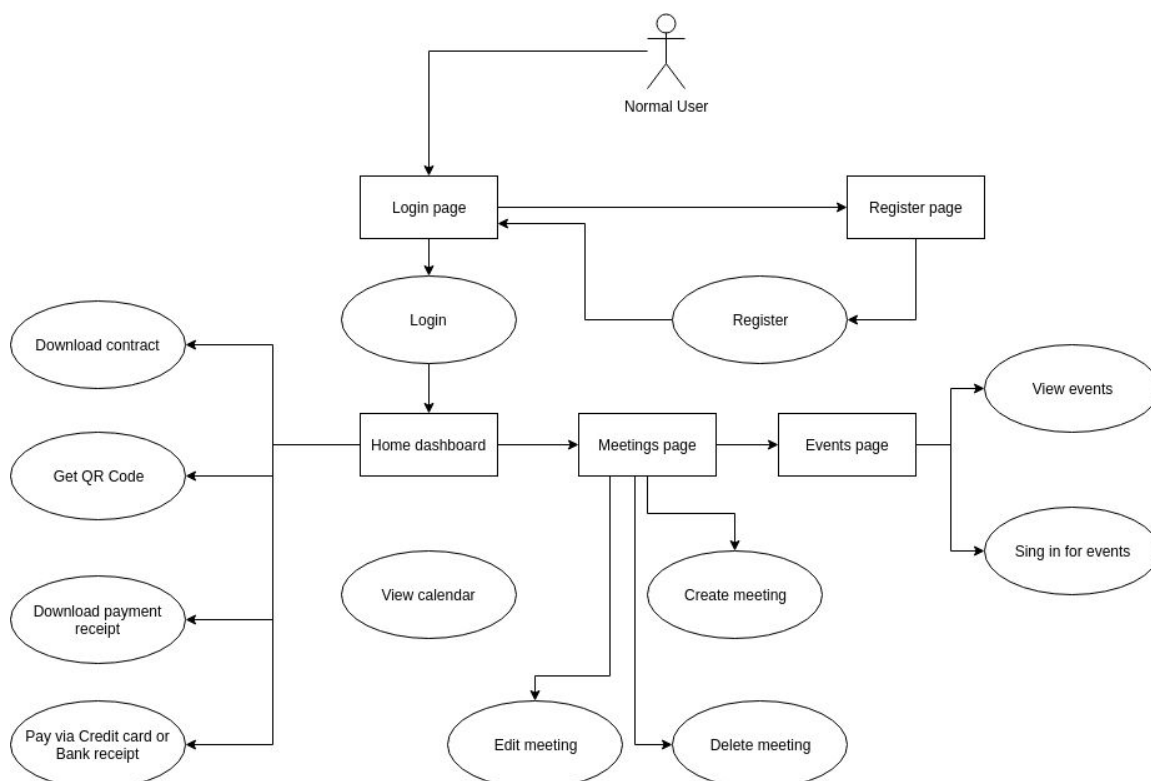


Figura 23 - Diagrama use-case pentru utilizatorii normali

### III. 2 Utilizatori administrativi

Deși aplicația are trei tipuri de conturi pentru personalul administrativ, toate acestea utilizează același panou de administrare, însă în funcție de tipul contului, aceștia au restricționate sau nu anumite părți ale panoului. Restricțiile sunt astfel:

- Administrator: acces global, nu are nici o restricție.
- Secretar: Poate edita conturile utilizatorilor normali, însă nu poate să le șteargă. Poate vizualiza plățile, să creeze evenimente și să vadă log-urile. Acesta nu poate să creeze alte conturi administrative.
- Contabil: Poate să vadă și să editeze plățile, iar pe restul panoului de administrare are accesul restricționat.

#### Secțiunea Dashboard

Odată ce utilizatorii se autentifică, aceștia vor fi redirecționați către panoul de administrare, către pagina cu utilizatorii:

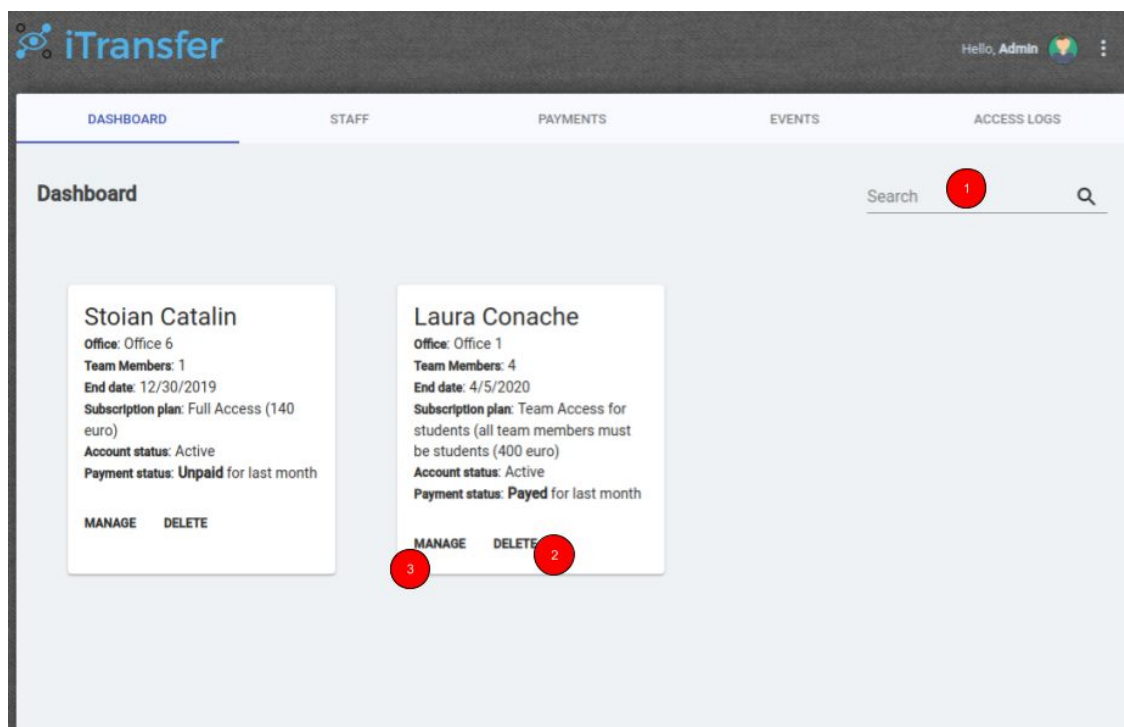


Figura 24 - Pagina administrare utilizatori

Din această pagină se pot vizualiza toți utilizatorii, aceștia vor fi afișați sub forma unui card ce conține informațiile relevante despre acel utilizator precum: Numele, biroul, numărul

de membrii în echipă, când se încheie contractul, subscription plan-ul, status-ul contului (activ / inactiv) și status-ul ultimei plăți generate.

Utilizatorii se pot filtre folosind câmpul "Search" (indicator 1 - Figura 23). Filtrarea utilizatorilor funcționează astfel: Odată introduse niște cuvinte cheie, algoritmul din spate va căuta în toate atributele contului, făcând potrivire între acestea și cuvintele cheie. Algoritmul va afișa doar acele conturi pentru care a existat cel puțin o potrivire pe unul din atribute. De exemplu dacă introduc cuvântul "Full Access", atunci vor fi afișate toate conturile care au activ un subscription plan de tipul *Full Access*. Este de ajuns doar să scriu cuvintele cheie iar algoritmul de căutare va rula automat, nu este necesară apăsarea vreunui buton sau tastă.

Pentru a accesa modalul de administrare a conturilor, trebuie apăsât butonul "Manage" (indicatorul 3 - Figura 23).

The screenshot shows a web application modal titled "Edit Laura Conache". It contains several sections: "General Information" with input fields for name, ID, GL, address, email, and role, plus a "Free account" checkbox (marked with a red circle 1); "Contract" with a "Download contract" link and a file upload area (marked with a red circle 2); "Team Members" with a list of names (Catalin, Alexandra, Maria) and a red circle 3; "Office" with a grid of colored squares representing office status (marked with a red circle 4); and a legend for "Taken", "Reserved", "Selected", and "Free" (marked with a red circle 5). At the bottom right are "CLOSE" and "SAVE" buttons.

Figura 25 - Modal administrare cont

Modalul din Figura 24 este împărțit în patru secțiuni diferite, fiecare fiind responsabilă după cum urmează:

- În prima secțiune numită "General Information" se pot modifica datele personale ale contului precum: Nume, cnp, serie, adresă, email și profil. Tot în această secțiune există un checkbox numit "Free account" (indicator 1 - Figura 24) destinat cazului în

care, din anumite motive, un cont poate să beneficieze de acces gratis în centru. Când timp acest checkbox este bifat, toate facturile viitoare care se vor genera pentru acest cont vor avea suma de plată egală cu 0 euro. De asemenea există și un câmp opțional pentru a scrie motivul pentru care acel cont a devenit "Free account".

- În a doua secțiune numită "Contract" (indicator 2 - Figura 24) se poate atașa contract-ul dintre centru și persoana/persoanele beneficiare. Odată cu atașarea contractului contul devine activ iar utilizatorul poate începe să folosească contul. Pentru a specifica data de expirare a contractului trebuie completat numărul de luni în care acesta este valabil. Acest număr se completează în câmpul "Number of months", sub eticheta "The contract expires in". Odată ce contul a fost activat, utilizatorul va primi o notificare prin email despre acest lucru.
- Secțiunea "Team members" (indicator 3 - Figura 24) este pentru a modifica numele membrilor de echipa pentru contul selectat.
- Din secțiunea "Office" (indicator 4 - Figura 24) se poate modifica spațiul de lucru alocat contului selectat, însă nu se poate selecta un birou deja ocupat sau rezervat. La fel ca și pe pagina de înregistrare, se regăsește o hartă a cu spațiile de lucru din interiorul centrului, facilitând astfel o mai bună vizualizare.

Pentru a salva modificările este necesară apăsarea butonului "Save" (indicator 5 - Figura 24), iar în caz că se dorește anularea modificărilor, trebuie apăsat butonul "Cancel". O singură excepție face atașarea contractului, care se salvează fără să fie apăsat nici un buton.

### Secțiunea Staff

Din această secțiune putem vizualiza, edita, crea sau șterge conturile cu rol de administrare.

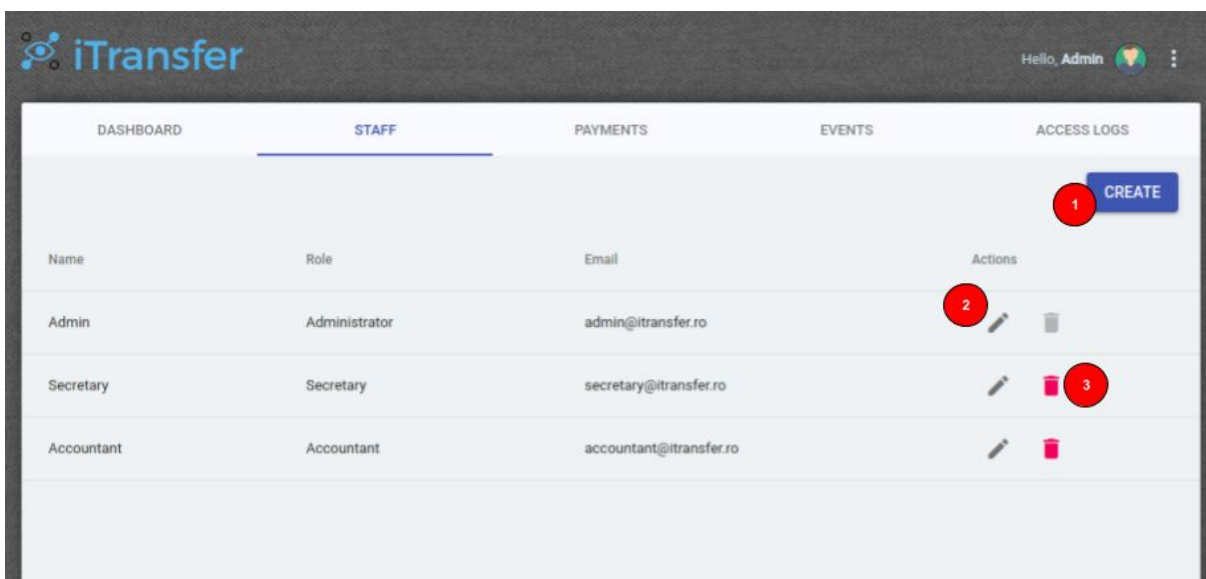


Figura 26 - Pagină pentru conturi administrative

Această pagină conține un tabel cu toate conturile cu drept de administrare. Un astfel de cont are un număr limitat de atribute, precum numele, rolul, email-ul și parola. Administratorul principal poate să creeze oricâte astfel de conturi dorește apăsând butonul „Create” (indicator 1 - Figura 25). Odată apăsat acest buton, se va deschide un modal unde i se cere email-ul, parola și rolul contului pe care dorește să îl creeze.

Figura 27 - Modal pentru creat/editat conturi cu drept de administrare

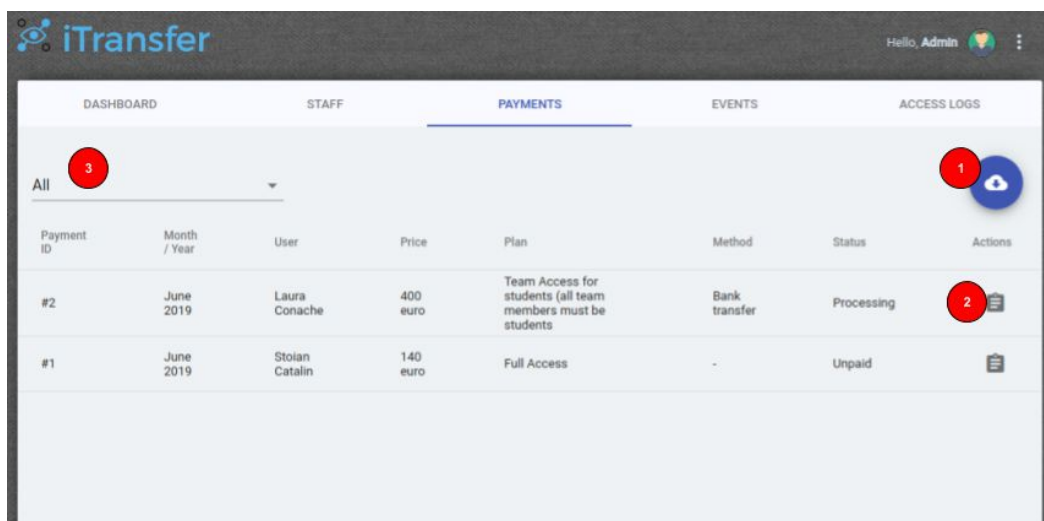
Un cont poate fi creat cu oricare dintre rolurile: Administrator, Secretar sau Contabil. Odată ce câmpurile au fost completate, contul se crează după apăsarea butonului "Save". Pentru a renunța la acest proces se apasă butonul "Cancel".

Pentru a edita un cont deja existent este necesară apăsarea butonului de edit (indicator 2 - Figura 25) din dreptul contului care se dorește să fie editat. Odată apăsat acest buton va apărea un modal identic cu cel din Figura 26. Pașii pentru editare sunt aceiași ca și pentru creare.

Pentru ștergerea unui cont se apasă butonul de ștergere (indicator 3 - Figura 25) iar apoi se confirmă acțiunea în modalul de confirmare. Odată ce un cont a fost șters, acesta nu va putea fi recuperat. Contul adminului principal nu se poate șterge.

### Secțiunea Payments

În această pagina se pot administra plățile utilizatorilor. Acestea vor fi afișate într-un tabel unde vor fi afișate cele mai importante informații despre plată precum: numărul unic de identificare a plății, luna și anul aferente, numele utilizatorului pentru care s-a generat plata, planul, metoda de plată în caz ca aceasta are alt status decât "Unpaid" și status-ul.



Payment ID	Month / Year	User	Price	Plan	Method	Status	Actions
#2	June 2019	Laura Conache	400 euro	Team Access for students (all team members must be students)	Bank transfer	Processing	[Actions]
#1	June 2019	Stoian Catalin	140 euro	Full Access	-	Unpaid	[Actions]

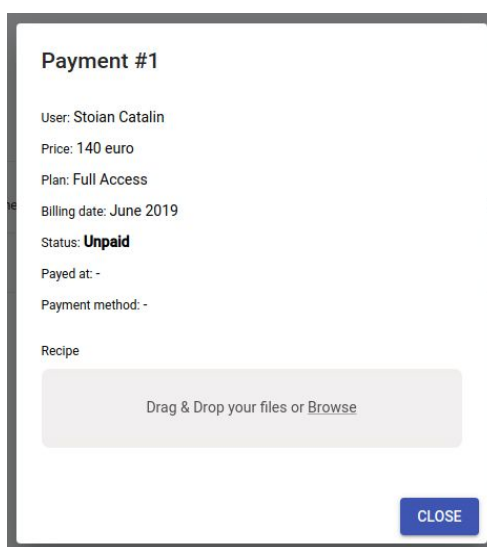
Figura 28 - Pagină pentru administrarea plăților

Deși în această pagină la început sunt afișate toate plățile tuturor utilizatorilor, avem posibilitatea să aplicăm filtre pe acestea pentru a vedea plățile doar unui anumit utilizator. Acest lucru se face modificând dropdown-ul din colțul stânga sus (indicator 3 - Figura 27). Ordinea acest plăți este descrescătoare în funcție de cea mai recentă plată, astfel încât cele mai recente să fie afișate primele.



Apăsând butonul din dreapta-sus (indicatorul 1 - Figura 27) se pot descărca plățile afișate în tabel în format CSV. Acest proces va ține cont de filtrarea aplicată pe tabel în momentul declanșării.

Pentru a vedea toate informațiile despre o anumită plată, se apasă butonul din coloana "Actions" de pe rândul aferent. Acest buton va deschide un modal unde va fi afișată plata detaliată.



**Payment #1**

User: Stoian Catalin  
Price: 140 euro  
Plan: Full Access  
Billing date: June 2019  
Status: **Unpaid**  
Payed at: -  
Payment method: -

Recipe

Drag & Drop your files or Browse

CLOSE

Figura 29 - Afișarea plății detaliate

În cazul în care utilizatorii normali efectuează plata la sediul iTransfer, personalul administrativ poate să schimbe status-ul plății în "Paid" atașând chitanța pe plata corespunzătoare în acest modal, în parte de jos a modalului, unde există eticheta "Recipe". Odată ce a fost atașată chitanța clientul va primi o notificare prin email că plata a fost achitată.

### Secțiunea Events

Din această pagină se pot crea, vizualiza, edita sau șterge evenimente precum conferințe, debate-uri sau prezentări cu interes tehnologic. Aceste evenimente vor avea ca și attribute: numele evenimentului, data începerii și data terminării, locația și prețul. În cazul în care prețul este 0 atunci în tabela va apărea sub label-ul "Free".

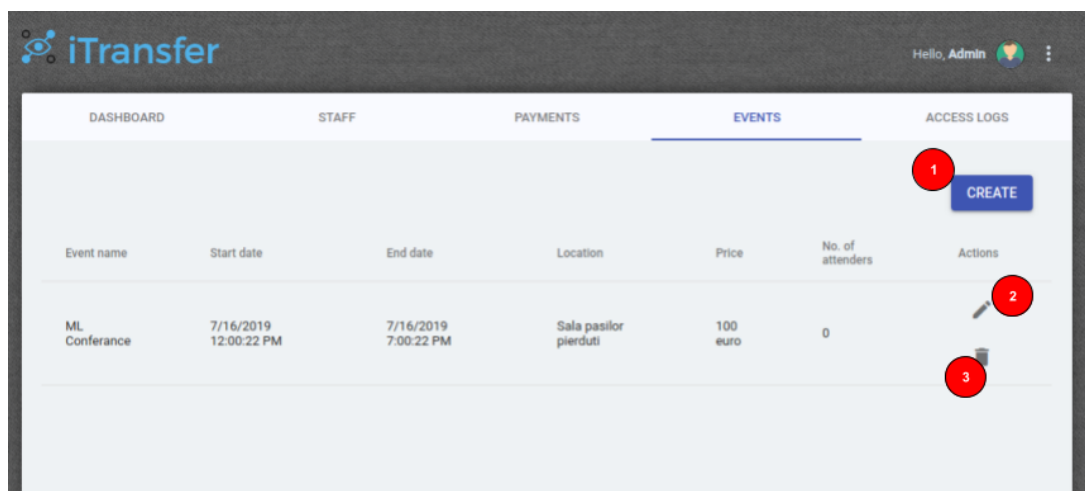


Figura 30 - Pagină pentru administrarea evenimentelor din cadrul centrului iTransfer

Figura 31 - Modal pentru creare / editare evenimente

Pentru a crea un eveniment trebuie apăsat butonul ”Create” (indicator 1 - Figura 29). Acest buton va deschide un modal (Figura 30) unde se va completa cu datele evenimentului. Odată ce S-au completat aceste date, este necesară apăsarea butonului ”Save” pentru a duce procesul la final.

Pentru editarea unui eveniment trebuie apăsat butonul de editare din dreptul evenimentului (indicator 2 - Figura 29). Acesta va deschide același modal ca și la procesul de creare, doar că câmpurile sunt deja completate. Pentru a finaliza procesul de editare se apasă butonul ”Save”.

### Secțiunea Access Logs

Pe această pagină sunt vizibile toate intrările clienților în interiorul centrului iTransfer. Sunt afișate în ordine descrescătoare în funcție de dată.

User name	Office	Access date
Stoian Catalin	Office 6	6/3/2019 1:27:01 PM
Stoian Catalin	Office 6	6/11/2019 8:31:37 PM

Figura 32 - Pagină pentru vizualizarea intrărilor în centru

Există posibilitatea ca aceste log-uri să fie filtrate după zi, sau după anumite cuvinte cheie, care la fel ca pe pagina de utilizatori, se poate căuta după anumite cuvinte cheie.

Log-urile vor fi afișate astfel: Utilizatorul care a intrat în clădire, biroul la care a mers și data exactă când s-a întâmplat.

Pentru accesul în clădire utilizatorul trebuie să scaneze QR-code-ul prezent în prima pagină (Figura 16).

Pentru a ilustra cât mai exact cazurile descrise mai sus, dar totodată și pentru a face o sintetizare a celor discutate, putem consulta diagrama use-case pentru acest timp de utilizatori:

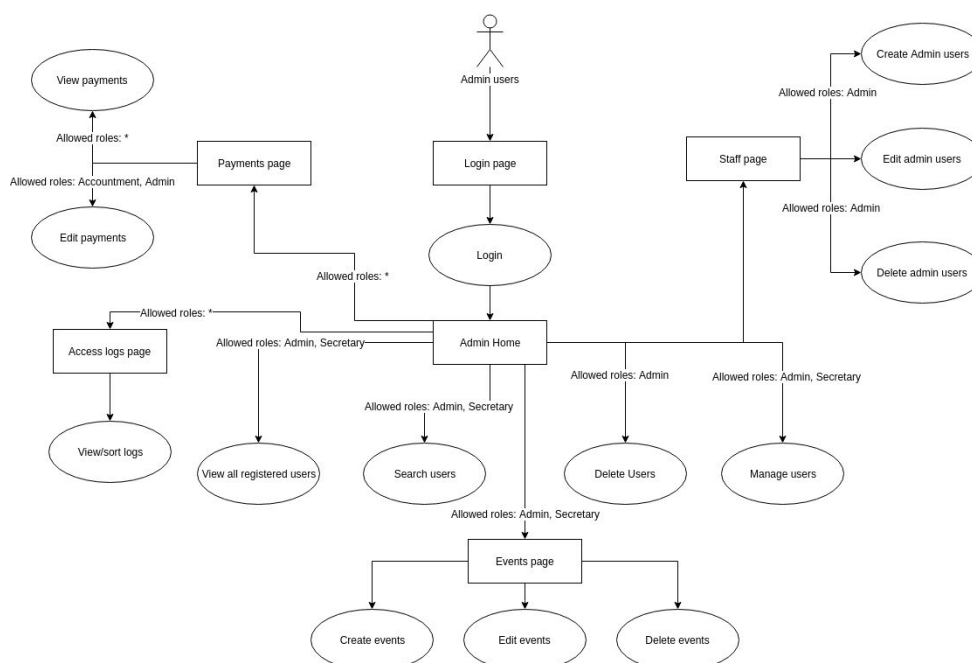


Figura 33 - Diagrama use-case pentru utilizatorii administratori

### III. 3 Alte cazuri de utilizare

Un caz comun regăsit pentru orice tip de cont, este modificarea profilului și a datelor personale. Această operațiune se face navigând către pagina profilului de utilizator. Acest lucru este posibil apăsând butonul "Profile" (indicator 1 - Figura 32) și imediat se afișează pagina profilului. Aici putem să facem modificările dorite iar pentru salvare apăsăm butonul "Save".

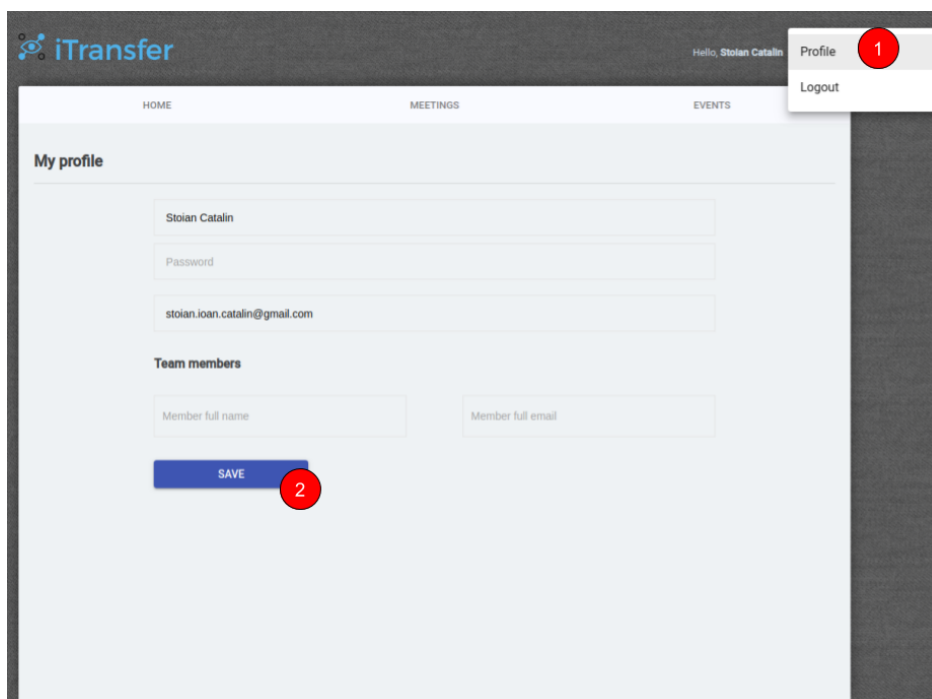


Figura 33 - Pagina pentru administrat profilul individual

Pentru intrarea în spațiile de lucru și monitorizarea lor, sunt puse la dispoziție două pagini spre servirea acestui scop. Prima pagină este cea conectată la device-ul unde utilizatorii vor scana codurile QR, iar a doua este o pagină ce va primi în timp real date despre cine scanează codurile QR. Această pagină va fi afișată în locurile de interes.

Pentru a funcționa prima pagină iar scanările să fie înregistrate prima dată trebuie arătat la cameră o parolă, sub forma unui QR-code. Odată ce parola a fost scanată, status-ul paginii va fi schimbat din "Denied" în "Granted". În acest fel putem ști când pagina este funcțională. Apoi utilizatorii pot intra în birourile lor doar scanând QR-code-ul la intrare. Odată ce un cod a fost scanat, atât pe paginile de vizualizare cât și pe pagina de scanare vor fi afișate datele utilizatorului.

Câmpurile afișate pe pagină sunt: nume, email, status-ul ultimei plății, dacă contul este sau nu activ (are semnat un contract), perioada de gratie (dacă este cazul), id-ul biroului unde trebuie să meargă și numele membrilor echipei.

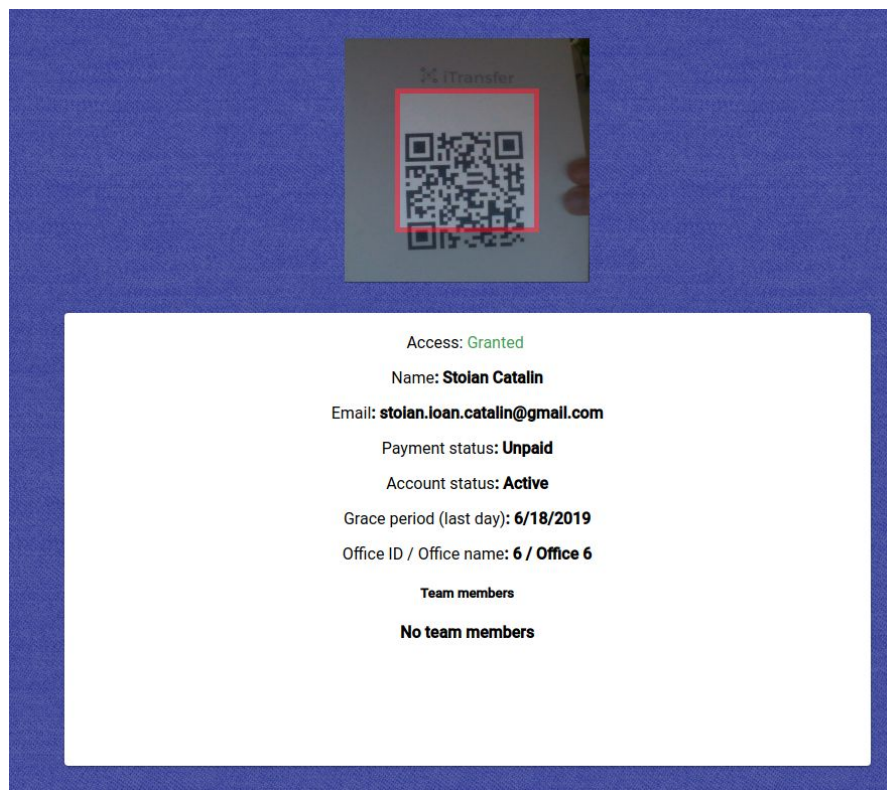


Figura 34 - Pagină pentru scanat coduri QR

## **IV. CONCLUZII**

### **IV. 1 Rezumat**

Consider că aplicația prezentată și-a atins scopul în a rezolva problema abordată, aceea de a oferi un mod inteligent și eficient de a administra centrului de transfer tehnologic iTransfer, precum și faptul că a fost realizată într-o manieră sustenabilă cu o arhitectură puternică, corect structurată, orientată către dezvoltare continuă. Astfel folosind tehnologiile precizate în Anexa1, am reușit să le folosesc cu succes cu scopul de a crea o aplicație web demnă să susțină sute de utilizatori atât prin aplicația server cât și prin aplicația vizuală. În ceea ce privește baza de date, consider că aplicația beneficiază de cea mai optimizată formă a ei, evitând duplicarea datelor, care oferă o structură ce interacționează foarte ușor cu aplicația server prin intermediul ORM-ului.

Faptul că aplicația are un portofoliu atât de mare de cazuri de utilizare, cred că își atinge cu succes scopul de a oferi o experiență cât mai ușoară a celor care trebuie să o folosească. Design-ul și experiența utilizatorului în timpul folosirii ei au fost atent gândite, ghidate după principii UI/UX, astfel încât să sugereze un prag de încredere oamenilor care o vor folosi. Un alt plus care vine să întărească afirmațiile anterioare este acela că aplicația se integrează cu GMail și GCalendar pentru a trimite email-uri pe post de notificări, respectiv pentru a crea/șterge evenimente în calendarele personale ale utilizatorilor, astfel ei beneficiază de caracteristici precum notificările de tip ”push to phone”. Un alt plus este acela că ofera un API public pentru a obține programul sălilor de evenimente și de ședințe.

De asemenea întreaga activitate a intrărilor în spațiile de lucru este salvată sub formă de log-uri, astfel personalul administrativ dispune de un întreg istoric.

### **IV. Direcții viitoare**

Ca și îmbunătățiri ce pot fi aduse aplicației ar acelea de a schimba scanarea QR pentru accesul în spațiile de lucru cu scanare NFC. Evident pentru această schimbare ar fi necesară o investiție minimă în echipamente de citire NFC și pentru deschiderea automată a ușilor. O altă îmbunătățire ar fi oferirea posibilității ca utilizatorii să se poată autentifica prin platformele de socializare precum Facebook, Instagram sau LinkedIn. Iar nu în cele din urmă, una din cele mai interesante îmbunătățiri ar fi construirea și integrarea unui asistent virtual în interiorul aplicației. Acest asistent să beneficieze de inteligență artificială și să poată ajuta persoanele cu dezabilități să folosească aplicația la fel de ușor ca orice altă persoană.

## V. BIBLIOGRAFIE

- [1] React Documentation, <https://reactjs.org/docs/getting-started.html>
- [2] Material UI, <https://material-ui.com/getting-started/installation/>
- [3] Redux, <https://redux.js.org/introduction/getting-started>
- [4] NodeJS, <https://nodejs.org/en/docs/>
- [5] KoaJS, <https://koajs.com/#>
- [6] What is a RESTful API?, Iazlojuly (Oct 27, 2016),  
<https://medium.com/@lazlojuly/what-is-a-restful-api-fabb8dc2afeb>
- [7] QR Code scanner, <https://www.npmjs.com/package/react-qr-reader>
- [8] 5 easy steps to understanding json web tokens (jwt), Mikey Steacky-Efantis (May 16, 2016)  
<https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec>
- [9] QR Code, <https://www.npmjs.com/package/qrcode.react>
- [10] Docker, <https://docs.docker.com/>
- [11] React File Pond, <https://www.npmjs.com/package/react-filepond>
- [12] React Credit Card, <https://www.npmjs.com/package/react-credit-cards>
- [13] Sequelize, <http://docs.sequelizejs.com/>
- [14] Google Calendar and GMail API, <https://developers.google.com/docs/api/>
- [15] UI Design Guide 2018 - Rules of User Interface Design, Abhishek Kumar (Jun 3, 2018)  
<https://medium.com/sodio-tech/ui-design-guide-2018-rules-of-user-interface-design-6307c24cd87c>
- [16] Googleapis, <https://www.npmjs.com/package/googleapis>

## VI. ANEXA 1. TEHNOLOGII UTILIZATE

Am încercat să aleg stack-ul tehnologic în așa manieră încât să facă codul ușor de întreținut și să se plieze pe nevoile aplicației pentru a face posibilă dezvoltarea tuturor caracteristicilor specifice acesteia.

Astfel, pentru aplicația vizuală am ales să folosesc ca și tehnologie de baza React combinat cu Redux.

**React** este o bibliotecă pentru a construi interfețe web de tipul ”single-page application”. Această librărie face ca crearea de componente UI să fie relativ simplă, apropiindu-se foarte mult de conceptul de web components.

Web Components este un concept relativ nou, ce a câștigat teren în toate comunitățile de FrontEnd development în ultimii ani. Acesta este defapt un API ce permite programatorilor să creeze componente web customizate, reutilizabile și încapsulate. Acest lucru înseamnă că dacă ai o anumită componentă pe o aplicație poți să o folosești în orice altă aplicație, indiferent de tehnologiile utilizate pe cele două, venind fără nici o dependență.

**Redux** este o bibliotecă pentru menținerea și administrarea stării aplicației în cadrul căreia se folosește. Astfel librăria are 3 concepte relativ simple: Acțiune, Reducer, Store. Acțiunea este un eveniment emis de acțiunile utilizatorului final făcute pe aplicație, cum ar fi apăsarea unui buton sau inserarea unui text. Reducer-ul este cel care decide cum se modifică starea în funcție de acțiune. O aplicație poate să aibe unul sau mai multe astfel de reduceri, depinde cum dorește programatorul să își structureze aplicația în funcție de nevoi. Store-ul este un ”pod” între starea finală și aplicată a aplicației și starea teoretică, produsă de reduceri.

Pentru componentele vizuale am folosit **Material UI**. Am ales să fac acest lucru pentru a nu reinventa roata de fiecare dată când trebuia să dezvolt anumite componente. Această bibliotecă îți pune la dispoziție o suită generală de componente deja create în React. În acest fel eu am putut să mă concentrez doar pe personalizarea aplicației conform design-ului gândit de mine, și mai puțin pe partea muncitorească.

Pentru aplicația server, am ales să folosesc framework-ul KoaJS, Socket.IO și Sequelize.

**KoaJS** este un framework creat în limbajul Node.js, scris într-o manieră care să ofere programatorului o experiență ușoară și să nu se concentreze decât pe acele părțile care chiar contează. Codebase-ul framework-ului este de ~570 SLOC, deci este un framework de



dimensiuni relativ mici. Acesta funcționează pe bază de middleware-uri înlănțuite, punând la dispoziție două stream-uri: request, respectiv response. Așa după cum le este și denumirea, unul dintre ele reprezintă datele request-ului iar celelalte cele ale response-ului. Viziunea framework-ului este ca aceste 2 stream-uri să fie alterate de middleware-uri, obținând astfel rezultatele dorite. Acest framework funcționează folosind protocolul HTTP/2 ([RFC7540](#)), iar în cazul în care acesta nu este suportat, atunci se face fallback la HTTP/1.1 ([RFC2616](#)).

**Socket.io** este o librărie ce se folosește de protocolul web-sockets ([RFC6455](#)) pentru a trimite informații în mod bidirecțional. Cazul concret în care s-a folosit această tehnologie în aplicația prezentată este acela ca anumite evenimente produse de scanarea codului QR trebuie să ajungă pe mai mulți clienți prestabiliți, mai exact pe ecranele afișate în birourile secretarilor. Dacă browser-ul nu suportă protocolul menționat anterior, atunci librăria va face fallback automat la o metodă numită long-pulling. Aceasta este un algoritm folosit înainte ca web-socket-urile să fie suportate sau să aibă o arie de browsere atât de largă de suport. Algoritmul presupune apelarea unui endpoint și menținerea acestuia în așteptare până la expirare sau până primește un răspuns din altă parte. După oricare din aceste 2 situații, procesul se reia de la început.

Sequelize este o librărie de tipul ORM orientată pe promise-uri. Un promise este un obiect ce reprezintă un eventual succes (sau eșuare) a unei operații asincrone, care returnează rezultatul acelei operații. ORM (sau object-relational mapping) este procesul prin care baza de date este reflectată de obiecte (din programarea orientată pe obiect) pentru a reprezenta datele ce le are stocate. Acest proces ușurează prelucrarea datelor și oferă un confort programatorului. Dezavantajul ORM-urilor este că nu îți permite să faci query-uri foarte complexe, însă acest neajuns poate să fie reprimat de posibilitatea de a executa query-uri direct în baza de date.

Pentru configurarea mediului în care aplicația funcționează am folosit **Docker**. Docker este un set de aplicații service ca și serviciu ce oferă posibilitatea de a încapsula în containere mediul unei aplicații, cu tot ce are nevoie. Un container este un mini sistem de operare, independent și executabil unde se pot instala și configura mediile de viață a software-urilor, după care se acestea se pot muta de pe o mașină pe alta fără să fie necesară nici o altă configurare. Mediul de lucru a aplicațiilor practic va fi același indiferent pe ce mașina este instalat.

## VII. ANEXA 2. SECURITATE, AUTENTIFICARE ȘI QR CODE

Pe partea de securitate am ales să folosesc JWT combinat cu https și Docker. JWT (json-web-token) este o metodă de a face schimb de date într-un mod securizat, prin encriptarea anumitor chei, obținând aceste token-uri. Aceste token-uri sunt compuse din 3 părți:

- Header care conține algoritmul folosit pentru encriptare și tipul token-ului.

```
// example of jwt header
```

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

- Payload-ul care este corpul de informație a token-ului. Aici se memorează date importante precum id-ul utilizatorului a cărui token este.

```
// example of jwt payload
```

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}
```

- Verify signature este cheia encriptată pe aplicația server, cu o cheie privată. Doar cine are acea cheie poate să verifice dacă token-ul este autentic sau nu.

```
// example of function to create a verify signature
```

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
)
```

Https este un protocol fără de care JWT nu ar putea să funcționeze datorită faptului că prin HTTPS utilizatorii sunt asigurați că chiar dacă cineva le interceptează call-urile pe internet, acestea sunt encriptate și pot fi citite doar de destinatar.

Partea de securitate pe care o oferă Docker este aceea că dacă cineva găsește o cale de acces în sistem acesta va avea acces la datele site-ului (sau a container-ului în care a pătruns, însă nu va putea să treacă mai departe, accesând restul rețelei.

Codul QR este o metodă ce funcționează exact ca și codul de bare din supermarket-uri. Acesta folosește un tabel de informație și în funcție de locul unde pătratul este colorat cu negru, acesta se uită în tabel la poziția respectivă și înlocuiește cu caracterul aferent.

## **VIII. Anexa 3. Integrare cu Google API**

Google pune la dispoziție un API foarte generos, cu posibilități nemărginite ce au fost implementate și integrate cu multe aplicații. Pentru aplicația prezentată în această lucrare de licență am ales să folosesc doar două dintre API-urile google, și anume Google Mail și Google Calendar.

Înainte de toate trebuie să preciez că indiferent de API-ul folosit, trebuie urmați anumiți pași și anumite configurări pentru a avea o comunicare sigură între aplicația dezvoltată și Google. Acești pași includ crearea unui cont de tipul sistem. Acest cont este contul aplicației, cu care se loghează iar celelalte conturi cu care interacționează știu că este vorba de un cont de sistem. Securitatea este asemănătoare cu cea descrisă în Anexa 2, însă aici metoda se numește OAuth. Este o metodă mai avansată și mult mai sigură, însă motivul pentru care eu nu am implementat OAuth a fost că toate avantajele oferite de această metodă față de JWT nu sunt de folos pentru că aplicația nu este la o scară atât de largă precum Google.

Pentru a folosi API-ul de la google am folosit librăria „googleapis”. Aceasta este o librărie ce oferă deja toate aceste implementări de configurare, dar și implementări ale funcțiilor ce interacționează cu API-ul specific, precum creare de evenimente în google calendar sau trimiterea email-urilor prin GMail API.